# SaMaRis: un environnement pour l'expérimentation et l'étude du maintien des raisonnements

Jérôme Euzenat, Laurent Buisson

# S a M a R i s: un environnement pour l'expérimentation et l'étude du maintien des raisonnements

# S a M a R i s: a framework for studies and experiments on reasoning maintenance

*Jérôme Euzenat*
*IRIMAG/INPG*
*Laboratoire ARTEMIS*
*BP 53X*
*F-38041 GRENOBLE cedex*
*Jerome.Euzenat@sherpa.imag.fr*

*Laurent Buisson*
*CEMAGREF*
*Division Nivologie*
*BP 76*
*F-38402 SAINT MARTIN D'HÈRES*
*laurent@beaufort.grenet.fr*

SaMaRis est un logiciel destiné à l'étude et à l'expérimentation des systèmes de maintien du raisonnement, ou de tout autre type de systèmes tirant parti d'une représentation explicite d'un raisonnement afin de lui faire subir des opérations constructives (rétablissement de la cohérence), destructives (oubli) ou consultatives (explication). Son architecture est composée de quatre modules indépendants: le protocole de communication avec le système d'inférence, le graphe de dépendances représentant le raisonnement lui-même, les services associés au graphe et les applications générales sur ce graphe. SaMaRis n'a aucune connaissance de la sémantique associée au graphe par le système d'inférence, ainsi son action peut-elle être adaptée à divers types de raisonnements.

SaMaRis is an environment aimed at studying and experimenting reason maintenance systems (RMS) or such kind of systems that use an explicit representation of a reasoning in order to process constructive (belief revision), destructive (forgetting) or read-only (explanation) operations against it. Its architecture is made of a communication protocol with an inference system, a dependency graph which represents the reasoning, general algorithms (called services) associated to the graph, and general programs manipulating the graph (called applications). SaMaRis does not know any semantics given to the graph by the client and thus, it can be used with several kinds of reasoning systems.

Actes 8ième congrès AFCET-INRIA-ARC-AFIA «Reconnaissance des Formes et Intelligence Artificielle», Villeurbanne (FR), 25-29 novembre 1991, pp1233-1247 (AFCET, Paris (FR), ISBN 2-90-36-77-103-8)

L'un des aspects fondamentaux de l'intelligence artificielle est la déclarativité, c'est-à-dire la capacité à représenter explicitement et à manipuler toute la connaissance. En ce qui concerne le raisonnement, cet aspect est important et il est bon de disposer d'une représentation d'un raisonnement qui soit facilement manipulable. On trouve, dans les systèmes de maintien du raisonnement (SMR), une structure de graphe qui, si elle n'est pas universelle, se révèle extrêmement pratique et peut être utilisée dans de nombreuses applications de l'intelligence artificielle touchant au raisonnement.

On se propose, avec SaMaRis, d'exploiter cette structure de graphe de façon à ce qu'elle soit utilisable par différents systèmes. Le graphe peut être vu comme représentant l'état d'un raisonnement achevé ou en cours et peut être examiné (par exemple à des fins d'explication), modifié (à des fins de normalisation du raisonnement, de généralisation) ou utilisé pour raisonner sur le raisonnement. SaMaRis est un ensemble logiciel intégré et modulaire destiné à expérimenter:

- de nouvelles applications des graphes de dépendances (par exemple l'oubli),
- l'intégration de divers types de raisonnement existants (par exemple temporel, hypothétique ou incertain),
- la conception d'algorithmes de maintien du raisonnement efficaces et corrects.

Il est aussi un cadre pour des travaux plus théoriques tels que la démonstration de certains sous-ensembles de logiques non monotones ou méthodologiques comme l'étude de l'apport du SMR dans une application. Il n'est pas destiné au développeur d'applications mais bien à un concepteur de SMR ou de systèmes similaires; il pourra néanmoins être configuré par un développeur de systèmes de représentation de connaissance.

Après une rapide présentation des systèmes de maintien du raisonnement (§1), les principes de SaMaRis et son fonctionnement seront exposés (§2). Les composants sont détaillés par la suite: le graphe et les services (§3), le détail de l'application consacrée à la propagation (§4) et une autre concernant l'analyse du graphe de dépendances afin de mettre en évidence l'utilité d'un SMR (§5). Le protocole de communication est exposé au §6. Enfin, le §7 présente une utilisation de SaMaRis au travers de son interface avant de conclure sur les développements à venir.

# 1.Les systèmes de maintien du raisonnement

Le principe d'un SMR à propagation, ou TMS pour "truth maintenance system" [Doyle 79], est d'enregistrer, pour chaque inférence, la formule inférée au sein d'un *nœud* et de lui associer une *justification* représentant l'inférence. Une justification est un couple d'ensembles de nœuds, le premier ensemble étant l'ensemble des nœuds dont la présence dans la base est nécessaire à l'inférence (IN-liste) tandis que le second ensemble est celui des nœuds dont l'absence dans la base est nécessaire à l'inférence (OUT-liste). Les justifications ont donc la structure suivante: $<\{i_1,...i_n\}\ \{o_1,...o_m\}>$. L'ensemble des nœuds considérés par le système, associés à leurs justifications, forme un graphe orienté nommé graphe de dépendances (voir figure 1).

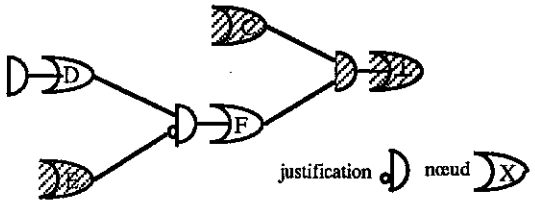


Figure 1. Le graphe de dépendances est représenté comme un circuit booléen dont les portes OU sont les nœuds et les portes ET les justifications dans lesquelles les nœuds de la IN-liste viennent directement alors que ceux de la OUT-liste passent par un inverseur. Le graphe représenté correspond aux justifications $<\{\}\{\}>:D$, $<\{D\}\{E\}>:F$ et $<\{C,F\}\{\}>:\perp$. Les nœuds (et justifications) en blanc sont considérés comme présents dans la base (ou valides) alors que ceux qui sont hachurés sont considérés comme absents (ou invalides).

Une justification est dite valide si tous les nœuds de sa IN-liste sont IN et tous ceux de sa OUT-liste sont OUT; un nœud, à son tour, est IN s'il a au moins une justification valide, dans le cas contraire, il est OUT. L'apparente circularité de ces deux définitions est évitée par les nœuds sans justification qui sont forcément OUT et les justifications ayant une IN- et OUT-liste vide qui sont forcément valides. Un étiquetage de chaque nœud du graphe comme IN ou OUT respectant ces contraintes est un étiquetage admissible; un étiquetage qui étiquette OUT tous les nœuds inconsistants est un étiquetage consistant. Le but de l'algorithme du TMS est de trouver un étiquetage faiblement fondé, c'est-à-dire un étiquetage admissible et consistant tel qu'un nœud ne puisse être étiqueté IN sur la base d'un argument circulaire.

Le TMS agit sur le graphe principalement quand une justification, correspondant à une inférence, lui est fournie. Il se charge alors de propager la validité nouvelle introduite par cette inférence au sein du graphe. Puis, si des faits contradictoires sont présents simultanément dans la base à la suite de cette inférence, le système invoque un mécanisme de

rétrogression[1] chargé de supprimer cette contradiction. La rétrogression choisit d'invalider l'une des hypothèses soutenant le nœud inconsistant. Une hypothèse est un nœud IN dont la justification supportante possède sa OUT-liste non vide, autrement dit, un nœud dont la présence est due à l'absence d'autres nœuds dans la base. Elle sera donc invalidée en ajoutant dans la base un des nœuds de cette seconde liste.

L'ATMS (pour "assumption-based TMS" [De Kleer 86]) ne considère que des inférences monotones (avec seulement la IN-liste: $<\{i_1,...i_n\}>$: c), mais il est capable de maintenir plusieurs contextes à la fois. Il considère des formules spéciales, appelées hypothèses, de façon à ce que l'utilisateur puisse configurer le contexte dans lequel il travaille. Un ensemble d'hypothèses est appelé un environnement et l'ensemble de tous les environnements possibles sur un ensemble d'hypothèses forme un treillis complet structuré par la relation d'inclusion (voir figure 2). Au lieu d'étiqueter les nœuds de manière absolue (avec des étiquettes IN ou OUT), chaque nœud est étiqueté par l'ensemble des environnements consistants sous lesquels il doit être présent dans la base. Un environnement est consistant s'il ne permet pas la présence du nœud $\perp$ et les environnements placés dans les étiquettes sont minimaux dans le sens où ils ne sont pas comparables. Après chaque inférence, le système calcule l'ensemble des environnements qui supportent l'inférence, l'insère dans l'étiquette du nœud inféré et le propage dans le graphe. Ainsi, pour savoir si une formule est présente dans la base, il suffit de comparer l'environnement caractérisant le contexte courant à ceux de son étiquette.

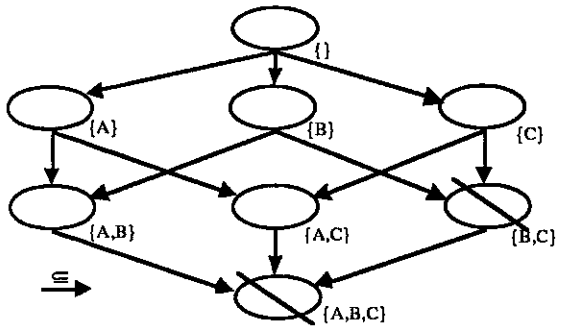


Figure 2. Le treillis des environnements construit à partir des hypothèses A, B et C dans lequel l'environnement {B, C} est inconsistant.

En résumé, le TMS prend en compte les inférences non monotones mais dans un seul contexte alors que l'ATMS est purement monotone mais prend en compte plusieurs contextes simultanément. On peut observer, par ailleurs, que les deux SMR doivent être connectés avec un système d'inférence qui leur communique ses inférences.

[1] Aussi appelé «retour-arrière».

# 2.Architecture de SaMaRis

Un système de maintien du raisonnement doit être aisément interfaçable avec différents mécanismes d'inférence. SaMaRis exploite cette première caractéristique en constituant un *serveur*, ou *mainteneur de raisonnement*, offrant diverses fonctionnalités à des *clients* qui sont des logiciels utilisant SaMaRis. Les clients sont en général des systèmes producteurs d'inférence, mais ils peuvent se contenter de consulter un raisonnement ou même de l'interpréter. Ainsi, la traditionnelle généricité des SMR est-elle préservée. Par ailleurs, la représentation interne des raisonnements est commune aux divers SMR, ou, du moins, il est possible de trouver une structure exploitable par tous. SaMaRis va donc réutiliser cette structure de graphe.

Les deux autres caractéristiques que va utiliser SaMaRis sont issues de l'architecture même des SMR. La figure 3 présente l'architecture d'un TMS classique telle que définie dans [Euzenat 88].

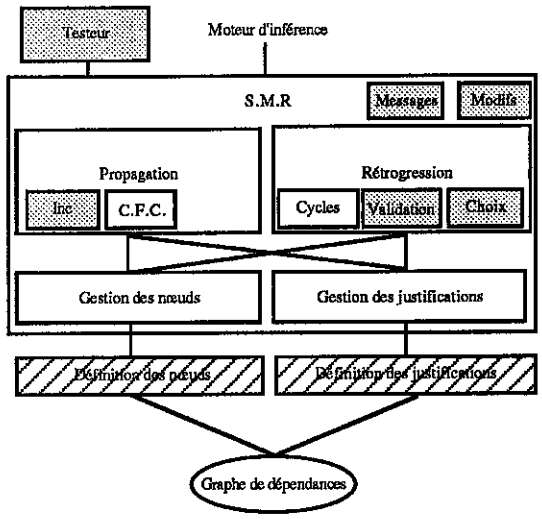


Figure 3. Architecture du système de maintenance de la vérité. Les modules légèrement grisés peuvent être modifiés pour diriger le comportement du système. Les modules hachurés sont à modifier pour un couplage fort (voir §3.2).

Tout d'abord la gestion des structures du graphe se fait dans des modules séparés de l'implémentation de ce graphe afin de ne pas en être tributaire. Ensuite, on remarque que des algorithmes tels que la détermination des composantes fortement connexes (CFC) ou la détection de cycles impairs sont des algorithmes généraux exploités par les SMR. Enfin,

la gestion des structures de données est utilisée par les deux modules du SMR (ici un TMS): propagation et rétrogression. SaMaRis va donc proposer non seulement une structure de graphe mais aussi des services qui sont des algorithmes généraux de manipulation de graphes et exploités par les divers SMR.

D'autre part, on constate que les SMR se décomposent facilement en modules indépendants (dans un TMS: propagation et rétrogression, dans un ATMS: propagation et rétablissement de la consistance...). Ces modules indépendants exploitant les services vont être intégrés dans la couche application de SaMaRis. Cela permet d'expérimenter diverses stratégies, d'intégrer divers systèmes (voir §5) et de rester indépendant des services offerts par la gestion du graphe. Le graphe n'est plus subordonné à une application particulière, mais est offert à toutes les applications possibles. SaMaRis permet donc de fédérer, autour de la représentation d'un raisonnement sous forme du graphe de dépendances, l'ensemble des services qui utilisent ce graphe: maintien du raisonnement, explication, oubli, propagation d'incertitude, de contextes, d'intervalles temporels... Ainsi, la charge que constitue, pour le système d'inférence, l'enregistrement des dépendances peut être compensée par la mise à disposition du graphe à tout un ensemble d'applications.

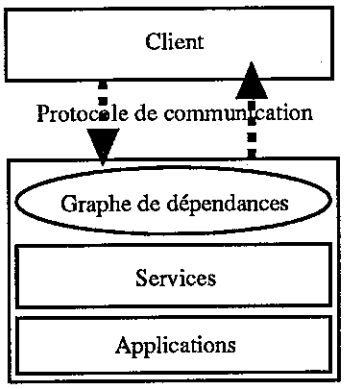


Figure 4. L'architecture de SaMaRis. On y retrouve les quatre composants: graphe, services, applications et protocole de communication permettant de communiquer avec le client. Les applications utilisent le graphe au travers des services communs.

L'objectif de cette architecture est d'être:

- modulaire: chaque module est défini par sa fonction et son interface avec les autres modules. Cela permet de pouvoir remplacer l'implémentation d'un module par une autre sans modifier le reste du système, ou même de se passer complètement d'un module.
- configurable: la fonction assurée par chaque module peut être configuré pour obtenir un comportement global du système.

En résumé, SaMaRis réorganise donc l'architecture des SMR (il s'agit de généraliser les structures présentes dans les SMR) et va donc mettre en jeu quatre types de composants:

1. Le *graphe* qui est une image du raisonnement produit.
2. Un *protocole de communication* entre SaMaRis et le monde extérieur (système d'inférence, interface graphique...).
3. Des *services* (propagation générique, décomposition en composantes fortement connexes, recherche de cycles, clôture et réduction transitive...) qui sont partagés par les applications et qui maintiennent le graphe de dépendances.
4. Des *applications* (révision des croyances, oubli, explication, propagation de coefficients de confiance, d'environnements, de valeurs de vérité, d'intervalles temporels...) qui exploitent le graphe de dépendances et font appel aux services.

Les composants de SaMaRis vont donc être détaillés en commençant par le graphe et les services (§3) et les applications (§4 et §5) avant de présenter le protocole de communication (§6). Au §7 une utilisation de SaMaRis est présentée au travers de son interface qui est considérée comme un client et peut donc être remplacée.

# 3.Graphe et services

## 3.1. Pourquoi un graphe?

Le graphe de dépendances peut être associé à l'ensemble des inférences produites. Le mode inférenciel est un modèle de calcul universel (on produit quelque chose à partir de ce qui existe) et, qui plus est, opératoire (il précise comment se fait le calcul et peut être implémenté tel quel). Par ailleurs, le caractère orienté du graphe de dépendances permet un compromis entre efficacité et complétude puisqu'il autorise des algorithmes concis et efficaces de propagation de marqueurs en son sein et, sous certaines conditions (de stratification en particulier), il permet de vérifier si une formule est dans une «extension» sans calculer toute celle-ci.

Il faut remarquer l'indépendance totale du graphe par rapport aux interprétations qui en sont faites:

- Les justifications peuvent représenter des implications de logique propositionnelle ou auto-épistémique [Reinfrank 90].
- Les justifications peuvent être engendrées par n'importe quelle règle d'inférence de logique classique (Modus ponens, résolution, règle de

déduction naturelle...) ou non monotone (règle de défaut [Junker& 90]).

C'est des multiples usages et interprétations qui peuvent être faits de son graphe que SaMaRis prétend tirer sa justification.

Bien sûr, considérer le graphe de dépendances comme une image de raisonnement est quelque peu réducteur. En particulier, le graphe s'apparente plutôt à une forêt de preuves qu'à un raisonnement qui est un processus d'essence dynamique. Cependant, il en capture certaines caractéristiques utilisables dans différentes applications.

### 3.2. La structure du graphe

Les graphes de SMR sont composés de sommets de deux types: nœuds (représentant les formules) et justifications (représentant les inférences). Chaque sommet renferme son nom et trois paquets d'attributs:

- Un état qui peut être complexe et composé, en fait, de plusieurs attributs. La sémantique de l'application est reflétée par l'état du nœud dont la structure est redéfinissable (voir §4); seules certaines applications ont accès à cet état.
- Des attributs de service qui sont utilisés par les services de manipulation de graphes offerts par SaMaRis (il s'agit, par exemple, de marqueurs permettant de ne pas boucler lors des parcours de graphe ou de rang d'un nœud). Ceux-ci ne doivent pas être modifiés, ils ne sont simplement pas déclarés quand le service correspondant n'est pas utilisé.
- Des attributs clients, permettant un couplage fort (voir §6.2), sont accédés par le client et ne sont jamais accessibles au services ni aux applications.

Le type des sommets du graphe peut donc être redéfini de manière à l'adapter à une utilisation particulière. Ces attributs sont gérés par les fonctions contenues dans le module de définition des sommets du graphe. Le graphe peut donc être regardé dans une «perspective objet» comme étant implémenté à l'aide de spécialisations particulières de classes de nœuds et de justifications dont la définition est surchargeable. Une telle implémentation permet de contrôler facilement la compatibilité des services, applications et nœuds.

### 3.3. Les services

Les services de SaMaRis permettent une plus grande efficacité des procédures qui manipulent le graphe en disposant en permanence de certaines structures et informations sur le graphe. Par ailleurs, les algorithmes proposés, étant indépendants des applications, peuvent être partagés par plusieurs d'entre elles. Les composantes fortement connexes (CFC) par exemple permettent de faire de la propagation de contraintes booléennes dans un ordre pertinent ou d'aller faire de la rétrogression directement sur des parties du graphe engendrant les extensions. Cela évite de voir coexister dans un même système plusieurs algorithmes presque semblables. Disposer d'un gestionnaire de réseau indépendant permet d'offrir un grand nombre de facilités aux applications. Certains de ces services étaient déjà utilisés par l'algorithme de TMS de Jon Doyle (clôture transitive), d'autres sont d'ores et déjà disponibles dans les systèmes que nous avons implémentés (CFC). La nouveauté provient de leur mise à disposition en tant que services; on peut en particulier citer:

- La construction et la maintenance de clôtures transitives [La Poutré& 87]. Celles-ci permettent de circonscrire les parties du graphe qui vont être explorées par le système.
- La recherche de cycles impairs.
- La construction et la maintenance des CFC [Aho& 74, Quintero 89]. Elle est utile pour de nombreux algorithmes de propagation mais est aussi utilisée par tous les autres services.
- La construction et la maintenance de stratifications [Pugin 88]. Moins précise que la construction de CFC, elle permet en particulier de travailler sur la maintenance d'extensions de prédicats et non plus seulement sur des propositions.
- La conservation de statistiques. Cela permet d'évaluer le travail fait par les algorithmes et donc de les comparer (SaMaRis est d'abord destiné à l'expérimentation de SMR). Elle permet aussi d'évaluer l'intérêt d'un SMR pour divers types de raisonnements (voir §7).

Ces algorithmes sont actuellement envisagés comme des algorithmes de maintenance incrémentale car ils devraient être efficaces en bénéficiant les uns des autres (les CFC permettent de maintenir les réductions transitives et de retrouver facilement les cycles; les clôtures permettent de maintenir les CFC...). Il est fort possible de les convertir en algorithmes paresseux si cela se révèle ne pas être le cas.

En résumé, SaMaRis offre une structure de représentation d'un raisonnement sous forme de graphe de dépendances et d'algorithmes généraux sur ce graphe appelés services. En ce sens il rempli bien son rôle de serveur de graphe. Il faut remarquer que si le graphe, par sa division en nœuds et justifications, représente l'ensemble des inférences d'un raisonnement, il reste suffisamment général pour autoriser plusieurs interprétations. En effet, si le raisonnement mené a toujours la même structure syntaxique, imprimée par les règles d'inférence utilisées, il peut être interprété différemment suivant diverses modalités: confiance accordée aux prémisses, nature hypothétique de celles-ci, probabilité d'occurrence des hypothèses, confiance dans les règles

d'inférence utilisées, période ou localisation d'occurrence des prémisses... L'interprétation du graphe est réalisée par les applications.

# 4.La propagation comme une application

C'est au travers des applications que SaMaRis s'instancie en un SMR ou un autre. Dans cette partie et la suivante, deux applications particulières sont envisagées: la propagation de valeurs dans le graphe et l'évaluation des ressources nécessaires à la production d'un ou plusieurs raisonnements.

La plus importante des applications est celle qui établit l'état (ou l'étiquette) des différents nœuds. Cela se fait, et c'est une constante des SMR, au moyen d'un algorithme de propagation. Cet algorithme propage des étiquettes au sein du graphe, en leur faisant subir certains traitements au passage des nœuds et des justifications. La propagation de données au sein d'un graphe de dépendances peut être généralisée à toute une classe de problèmes. Cette propagation est envisagée ici d'une manière totalement indépendante de son action.

## 4.1. Étiquettes et opérateurs

Pour instancier SaMaRis à l'aide du module de propagation, il faut définir ce qu'est l'état d'un nœud et comment il se propage dans le graphe. Pour cela, il faut déclarer un type de *jeton* qui sera propagé au sein du graphe, les valeurs initiales affectées aux nœuds et justifications, ainsi que trois opérateurs nommés concaténation (+), échantillonnage (×) et complémentation (~) à lui appliquer au passage des sommets du graphe. Encore une fois, les aspects méthodologiques introduits par la programmation objet (spécialisation et surcharge) permettent d'implémenter ceci aisément. Cette fois-ci, c'est le jeton qui est l'objet. Cette propagation permet alors de faire transiter:

- des étiquettes IN ou OUT (TMS),
- des environnements composés d'hypothèses (ATMS, CP-TMS [Euzenat 90]),
- des contraintes sur des variables (SHERLOCK),
- des mesures d'incertitude, d'imprécision, ou de probabilité,
- des intervalles temporels,
- des valeurs de vérité de différents types [Candelaria De Ram 90].

Il est possible de dresser le tableau des opérateurs mis en jeu pour diverses applications[2]:

| Système | type de valeur | nœud/just | (×) | (+) | (~) |
|---|---|---|---|---|---|
| TMS | booléen | ⊥/⊤ | $\wedge$ | $\vee$ | $\neg$ |
| ATMS | ens. d'ens. d'hypothèses | {}/{{}} | échantillonnage | $\cup$ | inutile |
| CP-TMS | contextes non monotones | {}/{[]} | $\cap_{CTXT}$ | $\cup_{CTXT=\cup}$ | $\neg_{CTXT}$ |
| flou | [0,1] | 0/1 | min | max | 1- (complément à 1) |
| probabilité | [0,1] | 0/1 | * | + | 1- (complément à 1) |

Il est clair qu'un des buts de SaMaRis est de permettre d'étudier la propagation simultanée de divers types de valeurs (elles sont alors intégrées au même jeton et les opérateurs sont combinés). Cependant, ce travail est fort complexe et doit être mené avec précautions.

## 4.2. Statut des étiquettes

L'étiquette apparaît jusqu'à présent comme une valeur de vérité associée à un nœud (ou des conditions d'attribution d'une valeur de vérité à un nœud). Le nœud est donc considéré comme un atome ou une proposition. Il est cependant possible d'étendre cette sémantique. Ainsi, le nœud peut devenir un attribut d'objet d'une représentation de connaissance par objets et l'étiquette une valeur ou un ensemble de valeurs d'attributs. Dans le même ordre d'idée, le système SHERLOCK [Cordier 87] manipule des formules partiellement instanciées et fait transiter dans le graphe les contraintes sur les variables qui permettent de satisfaire la formule. Le système VISILOG [Pugin 88] considère, quant à lui, les nœuds comme des symboles de prédicat et fait transiter dans le graphe les extensions de ces prédicats, c'est-à-dire l'ensemble des assignations possibles de ses arguments.

Il n'est pas certain qu'une telle approche donne des résultats corrects, cependant SaMaRis est l'outil adapté pour mener ce genre d'expérimentation. Un grand nombre de services de gestion du graphe y sont déjà disponibles et seule l'interprétation de la propagation changera.

[2] L'approche numérique évoquée dans ce tableau est très élémentaire puisqu'elle suppose l'indépendance des prémisses. L'intégration de techniques plus élaborées reste encore à faire. Pour ce qui est des valeurs de vérité, l'implémentation se fait à l'aide de tables de vérité.

### 4.3. Algorithmes de propagation

Comme il existe plusieurs interprétations des étiquettes, il est aussi possible de concevoir plusieurs algorithmes de propagation de ces étiquettes. Ainsi, l'algorithme de TMS de Jon Doyle utilise-t-il un sous-graphe du graphe de dépendances: le graphe de supports minimaux. Si cet algorithme est (parfois) plus efficace, il n'est pas du tout adapté pour une propagation qui en général doit être complète, comme dans l'ATMS par exemple.

SaMaRis doit donc disposer d'une série d'algorithmes de propagation différents suivant la façon de procéder. La propagation peut se faire nœud après nœud [Doyle 79], CFC après CFC [Goodwin 87] ou strate par strate [Pugin 88]. Qui plus est, la propagation peut s'appliquer à diverses opérations: une simple modification d'étiquette [Doyle 79], un remplacement d'étiquette [Buisson 90] ou une série d'opérations élémentaires (appelée transaction [Pugin 88]).

Pour l'expérimentation de ces techniques, SaMaRis est encore un outil important dans la comparaison des différents algorithmes. De plus, en utilisant une interface commune aux différentes procédures de propagation, il garantit la possibilité de substituer une technique à une autre si cela s'avérait nécessaire.

### 4.4. Autres applications

Si la propagation reste l'application privilégiée de SaMaRis, beaucoup d'autres sont à explorer qui bénéficieront aussi de l'architecture du système. Ainsi, une application dédiée à l'analyse du graphe de dépendances est présentée dans la partie suivante. On présente ici quelques autres pistes:

*Oubli*

L'oubli consiste en la suppression de prémisses ou d'hypothèses qui ont été introduites dans le graphe. Il s'agit de supprimer les nœuds correspondants tout en conservant la cohérence de l'étiquetage dans le graphe. L'oubli d'une prémisse peut se propager à toute une partie de graphe et ainsi entraîner la suppression d'autres nœuds et inférences; ces suppressions en cascade vont entraîner des suppressions dans la base du système d'inférence. Une sémantique de l'oubli a déjà été spécifiée pour certains systèmes [Strecker 91]. Un aspect intéressant de ce travail est certainement qu'il met en évidence le bien fondé de l'approche suivie par SaMaRis. En effet, le coût des SMR en terme de place occupée par le réseau de dépendances leur est souvent reproché. L'oubli permet de prendre en compte la nécessité de récupérer de la place en mémoire, non seulement au niveau de la représentation de connaissance, mais aussi à celui du graphe de dépendances. Ainsi, non seulement SaMaRis devrait être capable de gérer au mieux la place qu'il occupe, mais encore d'économiser celle occupée par la représentation de connaissance à laquelle il est associé.

*Rétablissement de la consistance*

La rétablissement de la consistance met à jour l'ensemble des étiquettes intervenant dans le graphe suite à la découverte d'une nouvelle inconsistance globale (on le trouve dans l'ATMS). Ceci est fait de telle sorte que chaque étiquette de chaque nœud ne prennent pas en considération les situations entraînant cette inconsistance.

*Rétrogression*

Suite à la découverte d'une inconsistance, le système remonte le graphe afin de retrouver les causes première de celle-ci. Il va ensuite ajouter, dans la base, une formule dont l'absence engendre l'inconsistance (TMS).

*Révision des croyances*

La révision des croyance fonctionne de la même façon que la rétrogression excepté qu'elle remet en cause certains *axiomes* suite à une inconsistance définitive découverte dans la base.

*Explication*

Les mécanismes d'explication des raisonnements produits utilisent toujours une représentation du raisonnement, soit qu'ils enregistrent leur propre représentation du raisonnement au cours de son développement, soit qu'ils reproduisent ce raisonnement pour l'analyser. SaMaRis est une structure d'accueil utile pour les systèmes d'explication du raisonnement. Tout d'abord parce qu'il est beaucoup plus pratique de synthétiser sur un graphe déjà construit que sur un graphe que l'on est en train de construire, mais surtout parce que SaMaRis offre déjà la possibilité de propager des informations à travers son graphe.

## 5. Analyse du graphe de dépendances

Dans [Buisson& 91a, b], on a présenté un premier travail concernant l'analyse des caractéristiques du graphe de dépendances et son utilisation dans l'utilisation d'un SMR ou d'un mécanisme de "caching" (ou enregistrement des résultats de chaque inférence). Ces résultats sont rappelés ici afin d'expliquer leur utilisation au sein de SaMaRis.

### 5.1. L'analyse du graphe

Ce qui est important pour les applications n'est pas tant la complexité au pire mais la complexité réelle du SMR quand il est confronté à un problème

réel. Dans cette perspective, le graphe de dépendances communiqué à SaMaRis représente le raisonnement réel effectué et peut être analysé afin de se rendre compte de l'efficacité du SMR.

Afin de donner des résultats précis, des hypothèses simplificatrices sur le graphe de dépendances ont été faites: (1) Il n'y a pas d'inférences non monotones. Ceci ne constitue pas une hypothèse très forte quand on admet la seconde. En fait, l'inférence non monotone dans un graphe sans cycle est un problème pour le moteur d'inférence mais pas pour le SMR. (2) Il n'y a pas de cycle dans le graphe. Cette hypothèse est très restrictive en théorie mais ne l'est pas dans de nombreuses applications. (3) L'analyse ci-dessous ne considère que des valeurs moyennes et suppose l'homogénéité du graphe. Cette fois-ci, par rapport à des applications réelles, l'hypothèse est forte. L'aspect général du raisonnement va être évalué et quantifié sur la base de valeurs moyennes en considérant que le graphe lui-même peut être décomposé en sous-graphes dans lesquels il est possible d'activer ou d'inhiber le maintien du raisonnement.

Quelques notations sont tout d'abord introduites: Soit B une base de connaissances consacrée à une application donnée. Nous considérons toutes les inférences lancées au cours d'une session «typique» de l'application; elles constituent le graphe de dépendances.

À l'instar de ce qui se passe dans un SMR classique, le graphe de dépendances ne représente pas toutes les inférences possibles sur B mais celles qui sont effectivement conduites. Les formules du graphe constituent l'ensemble F des formules du raisonnement (auxquelles correspondent les nœuds). N est le nombre de formules dans F. Dans F, on distingue les formules initiales (ensemble I) qui sont données et non inférées et les formules qui sont les objectifs des processus de raisonnement (ensemble Q).

Une chaîne est une séquence de formules et de justifications $f_0, j_1, f_1, \ldots j_n, f_n$ telle que pour tout $i \in [1,n]$, $f_{i-1}$ soit un antécédent de $j_i$ et $f_i$ soit le conséquent de $j_i$ dans le graphe. n est la longueur de la chaîne (c'est le nombre de justifications qu'elle contient).

La profondeur avant (df(f)) d'un nœud f est la longueur de la plus longue chaîne commençant à ce nœud (et terminant par un nœud de Q). La profondeur arrière (db(f)) du nœud f est la longueur de la plus longue chaîne (commençant par un nœud de I) et se terminant par f.

La largeur avant (wf(f)) d'un nœud f est le nombre de justifications auxquelles il est lié en tant qu'antécédent. La largeur arrière (wb(f)) d'un nœud f est le nombre de justifications auxquelles il est lié en tant que conséquent. wf(f) et wb(f) sont donc respectivement le nombre de connexions en avant et en arrière d'un nœud ou.

Si \ appliqué à deux ensembles représente l'ensemble des éléments qui sont dans le premier mais pas dans le second, wf est le nombre moyen de justifications dont une formule de F\Q est antécédente. wb pour sa part sera le nombre moyen de justifications dont une formule de F\I est conséquente. Ici, on considérera que wb=1 (cela signifie qu'une formule n'est inférée que d'une seule façon). Aussi,

$$wf = \frac{\sum_{f \in F\setminus Q} wf(f)}{|F\setminus Q|}, \qquad wb = \frac{\sum_{f \in F\setminus I} wb(f)}{|F\setminus I|}$$

$\rho$ est le rapport $\frac{|F\setminus Q|*wf}{|F\setminus I|*wb}$. Aussi, à cause de la valeur de wb, on aura $\rho = \frac{|F\setminus Q|*wf}{|F\setminus I|}$. Il s'agit du nombre moyen d'antécédents par justification dans le graphe de raisonnement.

Pour évaluer quantitativement l'impact du SMR, plusieurs constantes (qui dépendent de l'implémentation) sont distinguées:

- $\tau$inf: Le temps moyen pour une inférence quand toutes les prémisses sont connues.
- $\tau$rec: Le temps moyen pris pour enregistrer le résultat d'une inférence.
- $\tau$dep: Le temps moyen pris pour enregistrer une dépendance (qui représente une inférence).
- $\tau$sup: Le temps moyen nécessaire à la suppression d'une dépendance et d'une valeur enregistrée.

Une constante additionnelle, Treset, est le temps requis pour remettre l'application dans son état initial. Toutes ces valeurs devraient être aisément évaluées pour un raisonnement homogène.

L'action du caching et du SMR se résument en deux «effets de cône» évoqués ci-dessous avec les moyens de les évaluer. L'intérêt de l'utilisation de l'un et de l'autre réside dans un compromis entre les deux effets.

### 5.2. L'effet de cône arrière

Il y a un effet de cône arrière lorsqu'une connaissance est utilisée plusieurs fois au cours de l'inférence d'une autre. En d'autres termes, plus une connaissance est utilisée, plus le caching est intéressant. Cet effet est d'autant plus marqué que la connaissance est longue à inférer.

Avec le SMR, ces inférences intermédiaires ne sont plus réalisées qu'une seule fois. Pour la même raison, des inférences qui partagent les mêmes inférences intermédiaires bénéficient des résultats obtenus et enregistrés pour l'une d'entre elles (voir figure 5).

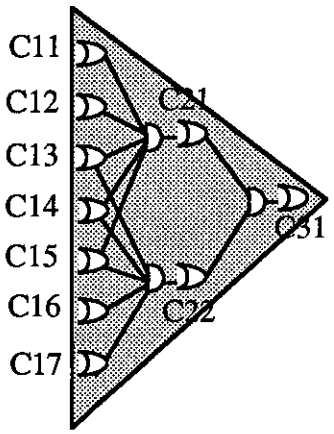


Figure 5. Afin d'obtenir C31, le système doit inférer C21 et C22 qui, à leur tour, nécessitent d'autres inférences. Ces inférences bénéficient du caching parce qu'elles partagent certaines de leurs inférences intermédiaires. Ceci explique pourquoi les inférences obtenues avec caching peuvent être plus rapides, même avant qu'elles ne soient enregistrées.

À première vue, le temps requis pour produire un raisonnement donné par un graphe est

$$TB = \tau inf * \sum_{f \in F\backslash} \nu(f)$$

où $\nu(f)$ représente le nombre de fois que f est utilisée durant une session. Il ne s'agit pas du nombre d'inférences dans lesquelles f apparaît (comme antécédent) mais le nombre de fois qu'il est nécessaire de produire cette inférence. Dans beaucoup d'applications, ce dernier est bien plus important.

Mais si l'on prend en considération la régularité supposée du raisonnement, on se rend compte que le temps pris pour inférer une formule (si $\rho \neq 1$) est:

$$TB(f) = \tau inf * \frac{\rho^{db(f)} - 1}{\rho - 1}$$

parce que $\frac{\rho^{db(f)} - 1}{\rho - 1}$ est la taille d'un arbre $\rho$-aire complet de profondeur db(f). Si $\rho=1$, alors TB(f)= $\tau$inf * db(f). Aussi, le temps total pour produire toutes les inférences d'une session (encore une fois, si aucun résultat d'inférence n'est enregistré et $\rho \neq 1$) est:

$$TB = \tau inf * \sum_{f \in Q} \frac{\rho^{db(f)} - 1}{\rho - 1}$$

Il faut noter que le quotient utilisé dans TB(f) est le nombre d'inférences dans le cône arrière. Si d'autres hypothèses sont prises en considération (un raisonnement non homogène, une structure autre qu'un arbre...), la formule peut être remplacée dans TB(f) par une autre exprimant le nombre d'inférences dans le cône.

Avec l'enregistrement de tous les résultats d'inférence, le temps nécessaire est:

$$TBcach = \sum_{f \in F\backslash} (\tau inf + \tau rec) = |F\backslash| * (\tau inf + \tau rec)$$

et

$$TBRMS = \sum_{f \in F\backslash} (\tau inf + \tau rec + \tau dep)$$

$$= |F\backslash| * (\tau inf + \tau rec + \tau dep)$$

Ainsi, le gain apporté est:

$$GB = \sum_{f \in F\backslash} [(\nu(f)-1) * \tau inf - (\tau rec + \tau dep)]$$

dans le cas du SMR, dans le cas du simple caching, la formule est la même sans la référence à $\tau$dep.

### 5.3. L'effet de cône avant

L'effet de cône avant s'énonce ainsi: plus la donnée est utilisée, plus l'invalidation est coûteuse. Comme dans le cas précédent, il y a effet de cône avant si une donnée est utilisée pour un grand nombre d'inférences. L'effet est négatif, il représente le travail à réaliser pour invalider un résultat enregistré.

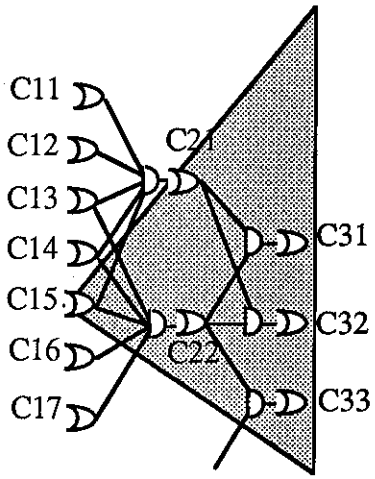


Figure 6. Le graphe représente les inférences produites par le moteur d'inférence. La partie en grisé est invalidée après la suppression de C15. Nous pouvons voir, qualitativement, que cette partie en grisé ressemble à un cône. Plus le cône est large, moins le SMR est intéressant parce que le nombre d'inférences à traiter est proche du nombre total d'inférences.

Un SMR est utile en cas d'invalidation (autrement, un caching brut tel que celui des systèmes en chaînage avant est suffisant). C'est l'invalidation

qui souffre de l'effet de cône avant. C'est cet effet qui va être évalué maintenant en considérant qu'une formule f de I est modifiée et que l'utilisateur du système a besoin des réponses au même ensemble de requêtes Q que précédemment. Dans le cas de l'enregistrement des résultats d'inférence, les réponses enregistrées ne sont plus valides. Avec un système de caching brut, le temps nécessaire à cela est:

$$TFcach = TBcach + Treset + TBcach$$

où, comme plus haut, $TBcach = |F\backslash I| * (\tau inf+\tau rec)$.

Avec le SMR, il est de:

$$TFRMS = TBRMS + Tinv + Treinf$$

avec, comme plus haut,

$TBRMS = |F\backslash I| * (\tau inf+\tau rec+\tau dep)$,

$Tinv = Ninval*\tau sup$,

$Treinf = Ninval*(\tau inf+\tau rec+\tau dep)$,

et $Ninval = \frac{wf^{df(f)+1} - 1}{wf - 1} - 1$ si $wf \neq 1$ et $df(f)$ sinon, c'est-à-dire, de nouveau, la taille d'un arbre wf-aire de profondeur df(f), c'est donc la taille du cône avant commençant au nœud f. Le facteur de branchement est wf parce que les justifications considérées n'ont qu'un seul conséquent, autrement, celui-ci aurait été multiplié par le nombre moyen de conséquents d'une justification.

Le gain apporté par le SMR est donc:

$GF = |N\backslash I|*(\tau inf+\tau rec-\tau dep) + Treset$

$- Ninval*(\tau inf+\tau rec+\tau dep+\tau sup)$

Si l'on oubli le temps d'initialisation et que $(\tau inf+\tau rec-\tau dep)*k = \tau inf+\tau rec+\tau dep+\tau sup$, le SMR sera avantageux quand $Ninval*k < N$ ce qui est vrai dans une application de taille importante.

### 5.4. Intégration à SaMaRis

Afin d'estimer l'utilité d'un caching ou d'un SMR, il faut connaître ces valeurs vis-à-vis de l'application (ainsi que le nombre de modifications escomptées...). SaMaRis est d'ores et déjà capable de procéder à cette analyse succincte du graphe réel. Si, de plus, on lui fourni les temps (associées à l'inférence, l'initialisation du système, l'enregistrement d'une justification...), il est capable de résumer les avantages et les inconvénients d'utiliser chacun des systèmes considérés. Ces deux étapes sont visibles sur la copie d'écran de la figure 7. Ainsi, l'utilisateur pourra connaître, dans le cas d'un raisonnement précis, les coûts et gains respectifs du caching et SMR.

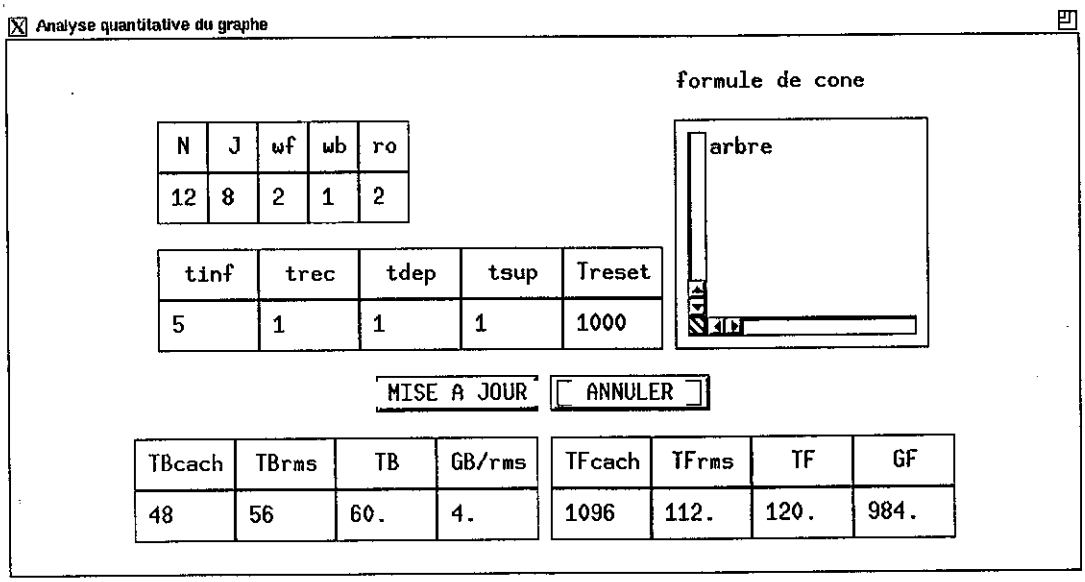


Figure 7. L'application d'analyse du graphe.

Après l'exposé de l'aspect serveur de graphe de SaMaRis et l'utilisation de ce graphe à l'aide des applications qui a été présenté, on montre comment communiquer avec SaMaRis.

## 6.Protocole de communication

Dans la perspective d'utiliser, avec différents systèmes d'inférence, les mêmes SMR, ceux-ci ont

été conçus d'une manière «générique», indépendamment de tout client. Cela conduit à la définition d'un protocole de communication entre systèmes d'inférence et SMR [MacAllester& 88, Maesano 89] et une méthodologie d'interfaçage [Euzenat 88]. Ainsi, la connexion entre un système d'inférence et SaMaRis est toujours possible et le temps de connexion est réduit de manière significative.

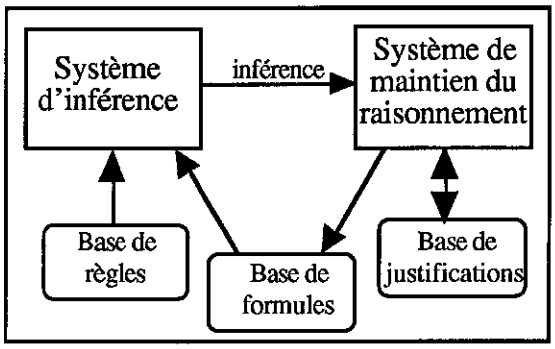


Figure 8. Interface générale entre le système d'inférence et le SMR.

Le cycle traditionnel des systèmes d'inférence s'appuie sur le contenu d'une base (de formules déduites ou faits) pour produire des inférences et intégrer le résultat des inférences dans celle-ci. Le SMR s'insère dans ce cycle en recevant, de la part du système d'inférence, une photographie de l'inférence et en mettant à jour la base des formules déduites. Le système d'inférence s'appuie à son tour sur cette base pour produire de nouvelles inférences.

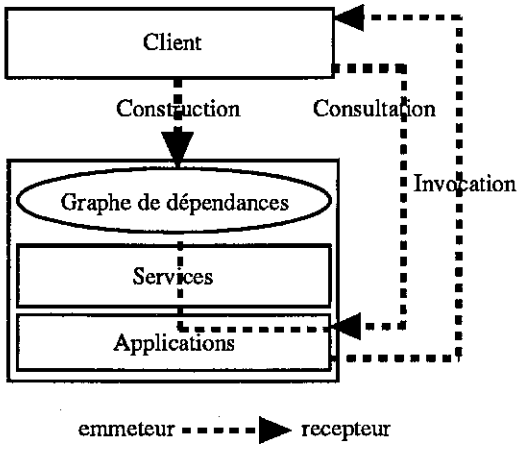


Figure 9. Les trois aspects du protocole de communication de SaMaRis.

Dans SaMaRis, ce protocole est décomposé de trois façons décrites dans la figure 9:

- Des primitives permettant la construction du graphe.
- Des primitives permettant sa consultation. Elles s'attachent aux informations disponibles au sein des nœuds ou fournies par les services. Mais elles peuvent aussi solliciter des applications pour avoir accès à des informations que celles-ci sont en mesure de fournir.
- Des mécanismes d'invocations qui permettent de faire appel depuis les applications de SaMaRis à des programmes fournis par le client.

### 6.1. Construction

Le premier aspect de la communication impose donc au système d'inférence de fournir au SMR tout ce qui doit y figurer: formules, inférences et contraintes. Le SMR ne manipule que deux sortes de structures: les nœuds représentant les formules et les justifications représentant les inférences. Toutes ces données lui sont fournies sous forme d'inférences; quand il est fait mention de formules dont il n'a jamais eu connaissance, le SMR construit automatiquement le nœud correspondant. Il est possible de construire un gestionnaire de base de formules déduites plus conséquent, mais le SMR devra toujours agir de la même façon.

Une inférence $i_1,\ldots i_n$, $NONo_1,\ldots NONo_m \Rightarrow f$ est fournie sous la forme d'une justification $<\{i_1,\ldots i_n\}\{o_1,\ldots o_m\}>$:f grâce à la primitive `sur-inference`:

```
(sur-inference (i1,…in) (o1,…om) f)
```

Une prémisse f (c'est-à-dire un fait présent dans la base sans avoir besoin d'être inféré) sera fournie au système de la façon suivante:

```
(sur-inference () () f)
```

Une contrainte signifiant qu'un ensemble de faits S ne peut pas être présent tout entier dans la base s'écrit sous la forme:

```
(sur-inference S () inconsistant).
```

Une seule primitive permet donc la communication des inférences produites par le système d'inférence au SMR. À partir de cette primitive, on en construit de plus haut-niveau:

`(addft <fait>)` permet d'introduire un fait dans la base. Celui-ci sera considéré comme une prémisse du raisonnement, c'est-à-dire un fait qui n'a pas besoin d'être inféré pour être valide.

`(inf (<fait>*) (<fait>*) <fait>)` transmet au SMR le résultat d'une inférence.

`(inconsistant <fait>*)` signale au SMR que la liste des faits fournie est inconsistante et que sa dérivation par le système doit déclencher le mécanisme de rétablissement de la consistance.

À partir de ces informations, le SMR va mettre à jour le graphe de dépendances et son étiquetage. Puis, le système d'inférence va pouvoir faire de nouvelles inférences ou l'utilisateur va pouvoir modifier les prémisses. Il existe donc quelques primitives supplémentaires:

`(init)` replace la base de faits dans son état initial.

`(supfait <fait>)` supprime, suivant des modalités précises [Strecker 91], la présence du fait dans le graphe.

`(addhyp <fait>)` permet de déclarer le fait comme une hypothèse dans des systèmes tels que l'ATMS.

### 6.2. Couplage fort / couplage faible

La liaison entre les deux modules est ainsi complètement spécifiée, mais il existe diverses manières de l'implémenter. Cela peut se faire par un couplage faible, où toute formule manipulée par le système d'inférence est dupliquée dans la base du SMR, ou par un couplage fort, dans lequel le système d'inférence considéré a directement accès à l'étiquette attribuée au nœud représentant sa formule. Il peut ainsi communiquer plus rapidement avec le SMR puisque celui-ci aura un accès direct aux nœuds du graphe alors que le couplage faible l'oblige à procéder à une recherche du nœud correspondant à la formule. D'un autre coté, le couplage faible assure l'indépendance des deux systèmes implémentés.

SaMaRis repose sur une méthodologie de couplage assez simple qui permet d'effectuer un couplage fort sans perdre cette indépendance. Elle consiste à permettre à l'utilisateur de redéfinir les structures de données concernant les nœuds et les justifications sans avoir à modifier les procédures qui accèdent à ce nœud depuis le SMR. Ceci est visible dans l'architecture proposée dans la figure 3 où l'implémentation des nœuds et des justifications se fait grâce aux modules amovibles présentés en hachuré. Il en va de même pour ce qui est du graphe de dépendances de SaMaRis (voir §4.2).

### 6.3. Consultation

La consultation, comme cela est visible sur la figure 9 se fait à partir de la couche application. Ceci permet, par exemple, d'avoir accès au résultat du travail effectué par les services sans se préoccuper de l'implémentation de ceux-ci (si les algorithmes maintiennent constamment leurs résultats où les recalculent à chaque appel). Ceci permet surtout, à partir de données brutes telles que celles enregistrées au sein des nœuds, de fournir des informations plus élaborées sans que ce travail revienne au client (qui alors ne disposerait plus de structures optimisées telles que les vecteurs de bits dans le cas des ATMS). La primitive principale pour l'interrogation des nœuds est `(etat <fait>)` qui rend l'état du fait (IN/OUT pour le TMS, l'étiquette pour l'ATMS, mais ce peut être une expression arbitrairement complexe).

L'exploitation de l'état par le client nécessite l'ajout de primitives d'affichage de l'état ainsi que de primitives de manipulations de celui-ci. L'application virtuelle qui sert à la consultation va fournir ces primitives de manipulation qui tirent parti de la valeur rendue par `etat`. Ainsi, pour l'ATMS on dispose de:

`(valide <fait> <environnement>)` qui indique si le fait est présent sous l'environnement.

`(consistant <environnement>)` qui signale si l'environnement est consistant.

`(contexte <environnement>)` qui fournit le contexte (c'est-à-dire l'ensemble des nœuds présents sous l'environnement).

`(maximal-consistant <environnement>)` qui donne l'ensemble des environnements consistants plus complets que l'environnement.

`(minimal-consistant <environnement>)` donne l'ensemble des environnements consistants moins complets que l'environnement.

Il faut remarquer que ces dernières primitives utilisent des calculs relativement complexes tirant parti de l'état global de la base. Les laisser au sein des applications permet de tirer parti des avantages offert par les services ce qui n'est pas le cas du côté du client qui dispose d'informations «décompilées».

### 6.4. Invocation

Enfin, la connexion des deux systèmes exige souvent la présence de modules spécifiques qui agissent en des points très précis du SMR. Ceux-ci sont facilement remplaçables et figurent en grisé dans la figure 3. Il s'agit, par exemple, des modules:

- `Messages` qui permet de spécifier la façon dont les messages s'affichent. Ceci est très utile pour construire une interface graphique.
- `Modifs` qui permet de réagir d'une façon appropriée quand un nœud change d'état. Ceci est très utile pour les couplages faibles où il faut mettre à jour la base propre au système d'inférence.
- `Validation` qui spécifie la réaction à adopter lorsque la rétrogression ajoute une justification.
- `Inc` qui permet de réagir de façon appropriée quand une inconsistance irréparable apparaît.
- `Choix` qui permet de choisir les hypothèses à supprimer de la base lors de la rétrogression. Diverses stratégies sont possibles, indépendantes du système d'inférence ou non.

Ce protocole, ainsi que la méthodologie de couplage qui a été mise au point, a pu être validé lors des expériences suivantes:

- Un couplage fort sur le logiciel IROISE, produit du CNET, qui possède les fonctionnalités d'OPS5

sans son réseau Rete. Le résultat obtenu met en œuvre des méthodes originales de rétrogression.

- Un couplage fort avec un réseau Rete qui a été intégré dans le logiciel Token (composé des modules Rete, Objets, TMS, ATMS et interface) de Cognitech.
- Un couplage faible avec le langage orienté-objet Kool de Bull. Le SMR est alors intégré en étendant simplement les objets de Kool à l'aide de nouveaux types d'attributs [Euzenat 89]. Les inférences par règles et objets sont maintenues. La connexion réalisée avec Kool a permis la liaison avec un démonstrateur du calcul des propositions utilisé comme un autre SMR.
- Un couplage faible avec l'interface graphique réactive de SaMaRis (voir §7) à laquelle sont connectés deux systèmes: le TMS et le CP-TMS.

Tous les SMR implémentés répondent à ce protocole et peuvent être facilement connectés à des moteurs d'inférence. De même, la plupart du code écrit est partagé par plusieurs systèmes permettant ainsi d'offrir plusieurs configurations de SMR. Ont ainsi été implémentées:

- Plusieurs stratégies de rétrogression.
- Plusieurs structures de justifications, de nœuds ou d'environnements.
- Plusieurs méthodes de réaction au changement d'étiquette d'un nœud.

# 7.La visualisation du graphe comme exemple de client

Le client privilégié de SaMaRis est, bien entendu, un système d'inférence. Celui-ci peut se concrétiser de multiples façons (représentation de connaissance générale munie de mécanismes d'inférence propres, moteur d'inférence spécialisé); l'interface de SaMaRis est elle-même considérée comme un client de SaMaRis. L'avantage de cette approche est que cette interface peut être supprimée ou remplacée.

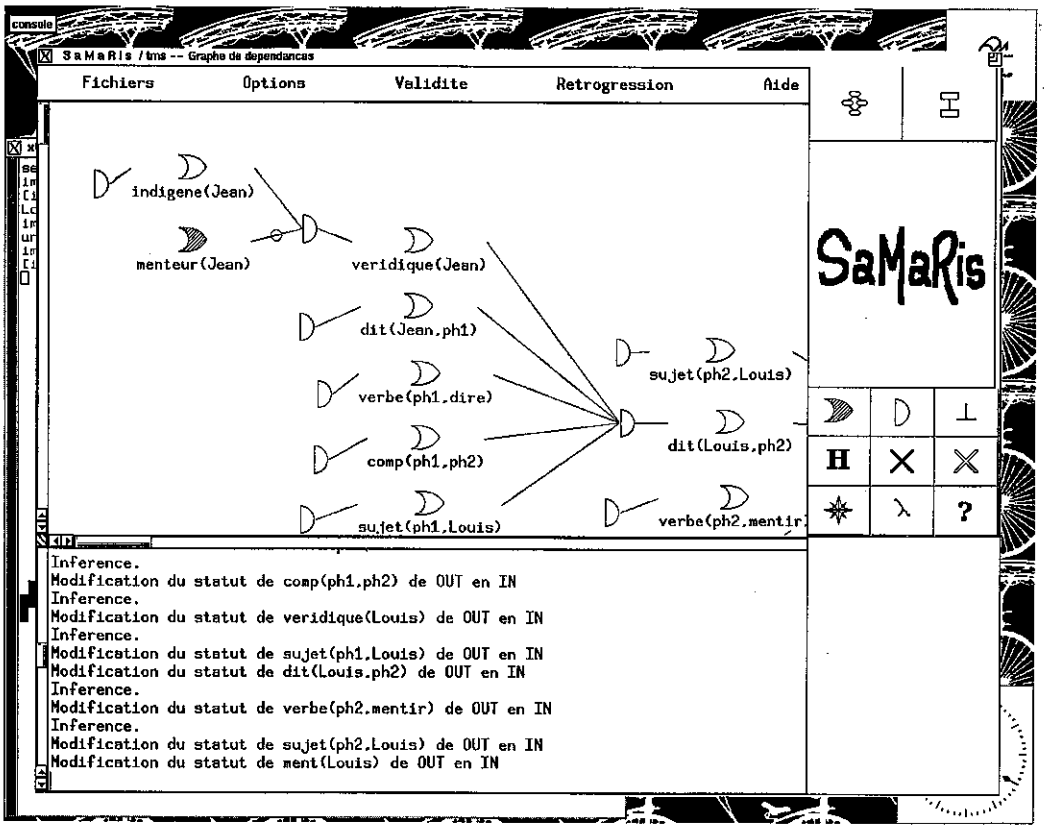


Figure 10. L'interface du système SaMaRis. La partie principale d'un écran est un éditeur de graphe qui permet d'exposer l'état des formules et leur dérivation. Une palette d'opérateurs sur le graphe permet de disposer de divers outils de manipulation du graphe. La barre de menus permet de paramètrer, à la fois, l'interface de SaMaRis et le SMR. Il est possible, par des commutateurs, de passer à l'édition d'autres graphes manipulés par le SMR. Enfin, la zone de communication permet d'afficher les attributs affectés aux nœuds ou d'interroger en Lisp le contenu de la base.

Le système SaMaRis possède une interface graphique fondée sur le principe de manipulation interactive des divers graphes exploités par les SMR [Euzenat 91]. Elle s'appuie sur le protocole de communication générique avec les SMR. Ceci permet d'utiliser la même interface (légèrement modulée en fonction du SMR) pour divers systèmes (TMS, ATMS, CP-TMS). L'intérêt de celle-ci est triple:

- Pédagogique: elle permet d'enseigner les principes des SMR par l'exemple. Le travail des SMR étant en effet très souvent souterrain, il est difficile de mettre en évidence leur action.
- Explicatif: l'interface peut se décomposer en modules incorporables dans une application. Elle permet alors de proposer la visualisation les graphes de dépendances du SMR comme des arbres de preuve.
- Démonstratif: elle permet au chercheur de tester le comportement de son système en lui soumettant des exemples de manière naturelle (en utilisant le symbolisme dans lequel il pense), mais aussi de démontrer à ses collègues les propriétés du SMR qu'il a construit.

Cette interface implémente un couplage faible (voir §3.2). Chaque action produite sur le graphe au moyen de la souris (ou d'un autre dispositif d'interaction) sera répercutée sur le graphe de dépendances et chaque modification dans l'étiquetage sera répercutée à l'écran.

L'interface de SaMaRis a été conçue de manière à représenter naturellement un raisonnement qui peut être complexe, et à être suffisamment instinctive et robuste pour pouvoir être manipulée par des novices. Elle doit permettre, à terme, d'accéder, non seulement à l'ensemble des services de SaMaRis, mais aussi à l'ensemble des applications utilisant le graphe de dépendances.

# 8.Perspectives

SaMaRis est actuellement développé sur Sun 3 et 4; il utilise les couches logicielles Unix (SunOS 4.0), X-Windows (X11R3), Lelisp (15.24), ainsi que quelques modules écrits en C. Son interface est développée en Aida (1.54) et fonctionne sur écran couleur ou noir et blanc. Il devrait être portable sur toutes les plate-formes sur lesquelles ces couches logicielles sont disponibles.

Les perspectives consistent actuellement en l'accomplissement du programme proposé ci-dessus. Les travaux en cours peuvent d'ores et déjà servir de base théorique à ce travail. Un certain nombre d'implémentations sont achevées:

- Le protocole de communication entre SaMaRis et l'extérieur est maintenant stable. Il a été utilisé dans diverses applications (voir §6) et concorde avec ce qui est couramment utilisé comme protocole [MacAllester& 88, Tayrac 90].
- SaMaRis possède une interface graphique très évoluée fonctionnant sur ce protocole.
- Deux SMR (TMS et CP-TMS, avec plusieurs variantes) sont disponibles qui partagent une partie de leur code. Il reste à séparer ce qui relève de l'application et ce qui relève du service dans ces systèmes.

La phase suivante consistera à développer des services plus performants autour du graphe et de nouvelles applications. Il importe, pour mener à bien le programme SaMaRis, de veiller à:

- L'intégration du réseau dans les représentations de connaissance existantes afin qu'il soit le plus efficace et économique possible.
- L'appréhension du réseau lui-même, sa maintenance et sa représentation de la façon la plus efficace qui soit.

Ces implémentations, faites dans un souci de modularité pourront tirer parti d'une implémentation sous forme d'objets. Par ailleurs, la sélection et l'intégration des modules devra pouvoir se faire de manière statique afin que la modularité ne nuise pas aux performances. Ces dernières réalisations ne présentent pas de difficultés d'ordre théorique ni pratique. Par contre, les développements de nouveaux SMR à partir de SaMaRis restent du domaine des recherches futures.

# 9.Conclusion

SaMaRis est un système destiné à répondre aux besoins d'analyse et de traitement d'un raisonnement produit de quelque manière que ce soit. Son but est donc d'être:

- Modulaire: les applications et services peuvent y être ajoutés ou retirés aisément.
- Générique: il est facilement connectable avec les clients qui doivent faire appel à lui (systèmes d'inférence, interfaces graphiques...).
- Efficace: SaMaRis œuvre en ce sens de deux façons; tout d'abord en partageant entre diverses applications des services qui seraient très coûteux s'ils n'étaient supportés que par une seule, ensuite en concevant des algorithmes plus performants pour ce qui concerne la propagation dans le graphe.

En un mot, SaMaRis doit être utilisable. Son architecture permet l'intégration et le partage de divers algorithmes généraux utilisés par les SMR. Ces services sont articulés autour d'une structure de données, le graphe de dépendances, représentant un raisonnement. Des applications fonctionnent en s'appuyant sur ce raisonnement. La décomposition des algorithmes classiques de SMR permet de rendre indépendantes certaines de leurs composantes et donc

de pouvoir modifier et assembler divers types de raisonnements.

La modularité de SaMaRis a deux effets: d'un point de vue expérimental, cela permet de remplacer facilement une implémentation d'un module par un autre; d'un point de vue applicatif, elle permet d'étendre le SMR utilisé vers d'autres utilisations sans repenser toute l'application.

Bien sûr, cette approche a ses limites, par exemple celle de ne pas autoriser à première vue un méta-raisonnement. En effet, le graphe étant isolé de sa sémantique (et n'ayant pas accès au contenu des nœuds) il est impossible qu'il renferme des auto-références. Si cela est une limitation de la puissance des raisonnements qu'il peut exprimer, c'est une assurance de simplicité.

# 10.Références


[Aho& 74] Alfred Aho, John Hopcroft, Jeffrey Ullman, **The design and analysis of computer algorithms**, Addison-Wesley, Reading, (MA), 1974

[Buisson 90] Laurent Buisson, **Le raisonnement spatial dans les systèmes à base de connaissances, application à l'analyse de sites avalancheux**, Thèse de l'université Joseph Fourier, Grenoble (FR), 1990

[Buisson& 91a] Laurent Buisson, Jérôme Euzenat, **Une analyse quantitative du raisonnement pour le maintien du raisonnement**, Rapport interne, Laboratoire ARTEMIS, Grenoble (FR), 1991

[Buisson& 91b] Laurent Buisson, Jérôme Euzenat, **A quantitative analysis of reasoning for RMSes**, 8th International symposium on methodologies for intelligent systems, poster session, Charlotte (NC US), 1991 à paraître

[Candelaria De Ram 90] Sylvia Candelaria De Ram, **Belief/knowledge dependency graphs with sensory groundings**, Actes 3ième symposium on artificial intelligence, pp103-110, Monterey (MX), 1990

[Cordier 87] Marie-Odile Cordier, **Unification contextuelle et raisonnement hypothétique: le moteur d'inférence SHERLOCK**, Actes 6ième RFIA, pp787-797, Antibes (FR), 1987

[De Kleer 86] Johan De Kleer, **An assumption-based TMS**, *Artificial Intelligence* 28-II, pp127-162, 1986

[Doyle 79] Jon Doyle, **A truth maintenance system**, *Artificial Intelligence* 12-III, pp231-272, 1979

[Euzenat 88] Jérôme Euzenat, **Un module TMS, version C0**, Rapport interne, Cognitech, Paris (FR), 1988

[Euzenat 89] Jérôme Euzenat, **Connexion Kool/RMS, spécifications**, Rapport interne Sachem JE004, CEDIAG/Bull, Louveciennes (FR), 1989

[Euzenat 90] Jérôme Euzenat, **Un système de maintenance de la vérité à propagation de contextes**, Thèse de l'université Joseph-Fourier, Grenoble (FR), 1990

[Euzenat 91] Jérôme Euzenat, **SaMaRis: visualiser et manipuler interactivement le raisonnement**, Actes 3ième convention intelligence artificielle, Paris (FR), pp219-238, 1991

[Goodwin 87] James Goodwin, **A theory and system for non monotonic reasoning**, *Linköping studies in science and technology* 165, 1987

[Junker& 90] Ulrich Junker, Kurt Konolige, **Computing the extensions of autoepistemic and default logics with a truth maintenance system**, Actes 8ième AAAI, pp278-285, Boston (MA US), 1990

[La Poutré& 87] J. La Poutré, J. Van Leeuwen, **Maintenance of transitive closures and transitive reductions of graphs**, *Lecture notes in computer science* 314, 1988

[Mac Allester& 88] David Mac Allester, Drew Mac Dermott, **What is a TMS ?**, Actes 7ième AAAI TMS tutorial, pp1-34, Saint Paul (MN US), 1988

[Maesano 89] Libero Maesano, **Spécifications fonctionnelles de l'interface programmatique RMS**, Rapport interne Sachem LM001, CEDIAG/Bull, Louveciennes (FR), 1989

[Pugin 88] Jean-Marc Pugin, **Contribution à l'étude des raisonnements non-monotones: le tableur logique**, Thèse de l'université Paris 7, Paris (FR), 1988

[Quintero 89] Jose Alejandro Quintero-Garcia, **Parallélisation de la maintenance de la vérité tirant parti des composantes fortement connexes**, Rapport de DEA, INPG, Grenoble (FR), 1989

[Reinfrank 90] Michael Reinfrank, **Fundamentals and logical foundations of truth maintenance**, *Linköping studies in science and technology* 221, 1989

[Strecker 91] Martin Strecker, **Formalisation de concepts d'oubli dans les systèmes de maintien du raisonnement**, Actes 8ième RFIA, ce volume, Lyon (FR), 1991

[Tayrac 90] Pierre Tayrac, **Étude de nouvelles stratégies de résolution, application à l'ATMS**, Thèse de l'université Paul Sabatier, Toulouse (FR), 1990